Autonomous Vision-Based Manipulation from a Rover Platform

Issa A.D. Nesnas Mark W. Maimone Hari Das

Abstract— **Current rover designs use on-board manipulators to enhance their capabilities for planetary exploration and in-situ science. In this paper, we describe how these manipulators can be used to perform two types of operations: rock sample acquisition for return to earth and instrument placement for in-situ science measurements. We describe the computational architecture, tools, and algorithms that we developed for this task. These algorithms integrate rover odometry, stereo visual tracking, and tactile sensing for each operation. We have successfully demonstrated these operations on a self-contained Mars Rover prototype, *Rocky 7*. We have demonstrated grasping a small rock sample from a distance of more than one meter away and placing an instrument on a boulder from a distance of more than five meters away.**

I. Introduction

FOLLOWING the success of the Sojourner Rover of the Mars Pathfinder mission, there has been an increased interest in adding manipulation on-board rovers to enhance their planetary exploration capabilities. Two types of manipulators have been used on several Mars rover prototypes: a mast that extends a stereo camera pair several feet above the rover's platform, and a manipulator arm that is used for sample acquisition, digging, and science experiments.

The mast's primary function is long range sensing of the surrounding terrain using narrow field-of-view cameras. To acquire a complete panorama of high resolution images, the mast must be able to pan and tilt its cameras. Taking advantage of these degrees of freedom, one or more science instruments are also mounted on the mast for placement onto a designated target. Because the mast carries sensitive sensors and instruments, it is not suitable for digging or acquiring rock samples. A second manipulator with shorter link lengths is used instead. This manipulator arm can also carry less sensitive science instruments.

Because of power consumption and mass constraints, these manipulators have limited degrees of freedom. In this work, we will demonstrate how we use these manipulators in conjunction with the vehicle's mobility system to compensate for their limited dexterity. We will also show how we use frequent visual feedback to compensate for the uncertainties and the simplified kinematic models of the system when tracking a target. Tactile sensing is used when the manipulators get close to the target.

In the next section, we briefly present some related work that uses sensor-based manipulation algorithms. In the sections that follow, we present our objectives, approach, and architecture. We also present our algorithms and provide some experimental results. We conclude with a summary and some planned enhancements to this work.

The authors are with the Jet Propulsion Laboratory, Pasadena, CA 91109, email: issa.nesnas@jpl.nasa.gov, tel: (818) 354-9709

II. Background

There have been several efforts in sensor-based manipulation especially in vision-guided manipulation. Some researchers developed algorithms to servo manipulators in the Cartesian space [1] [8], while others worked in the image plane using intensity-based approaches to vision-guided manipulation [10] [3]. Several researchers have achieved high frame-rate visual servoing [1] [8] [10] [3] [2] [7] [11]. However, most of this work assumed a dexterous manipulator mounted on a fixed platform. The relative size of the object in the images remained the same throughout the servoing process, and most of these efforts were demonstrated in an indoor environment where lighting can be controlled.

In our case, the manipulators are mounted on a moving rover platform. We rely on the mobility system to compensate for the limited dexterity of the manipulators. We compensate for uncertainties, resulting from the terrain roughness, and changes in the shape and size of the target by visually tracking the target as we approach it. Our rovers operate in an outdoor environment where specular reflections, shadows, and changes in lighting conditions make the vision problem quite challenging. Template searches on the intensity image can track well at long distances, but are less reliable at the final approach to the object [13]. To overcome these challenges, we use an elevation map rather than the raw image. The elevation map is generated from an on-board stereo vision system. We integrate the various sub-systems using a control architecture developed for this project to achieve intelligent autonomous behavior.

III. Objective

The objective of our project is to demonstrate intelligent manipulation on-board a rover platform subject to constraints similar to those encountered on Mars. In particular, our goal is to use a Mars rover prototype, *Rocky 7*, to autonomously pick rock samples selected from a distance of more that one meter away, and to autonomously place a science instrument on a rock designated from a distance of more than five meters away.

Without this level of autonomy, each objective would have taken more than five days to accomplish in a Mars mission. Tele-operation is very unlikely to succeed due to communication time-delay (several minutes for Mars) and a restricted communication window of a few minutes (twice per day for Sojourner during the 1997 Pathfinder mission). Alternatively, identifying the location of the target and then blindly driving toward it will not work either since

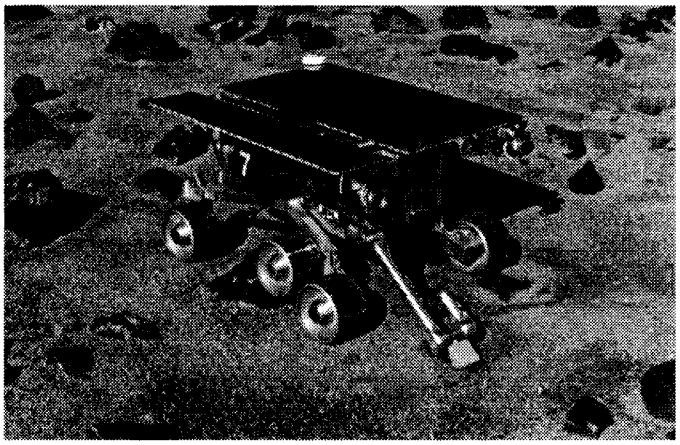

Fig. 1. The *Rocky 7* rover.

there are many disturbances and uncertainties in modeling rover motion over the terrain.

With the exception of the target selection, which is best accomplished by a human scientist, the rest of the operations for rock acquisition and instrument placement are completed autonomously without any intervention from a human operator. Next we describe the rover hardware and software control architectures that were developed and used to accomplish these tasks.

IV. System & Computing Architecture

A. The Rocky 7 System

Rocky 7 is a Mars rover prototype designed and built by the Long Range Science Rover team as a testbed for autonomous and intelligent algorithms [12] (Figure 1). *Rocky 7* is a six-wheel drive vehicle with a rocker-bogey mobility mechanism. It has two steerable front wheels and four non-steerable back wheels. The mobility mechanism defines the possible maneuvers the vehicle can perform.

Mounted onto the rover platform are two manipulators: a two degree-of-freedom (DOF) arm with two independently actuated scoops (making it an effective three DOF arm), and a three degree-of-freedom mast. The arm has a shoulder roll and a shoulder pitch, while the mast has an additional elbow pitch. Three pairs of stereo cameras are mounted on the rover. A narrow field-of-view (43° measured) stereo camera pair is mounted on the mast, and two wide field-of-view (103° measured) stereo camera pairs are mounted on the front and back sides of the vehicle. The latter pairs, also known as hazard avoidance cameras, are mounted at about 30 cm above the ground and are aimed downwards at a fixed 45° angle.

Due to the limited dexterity of the mast manipulator and the mounting of its stereo camera pair, the mast cameras cannot be used effectively for guiding the manipulator arm. So we rely on the hazard avoidance cameras when using the arm and on the mast cameras when using the mast.

B. Computing Architecture

The computing system consists of a 3U VME backplane with a 60 MHz 68060 processor with on-board Ethernet, two frame-grabbers, a digital I/O board, and an analog I/O board. The main processor runs a VxWorks 5.3 real-time operating system. Each actuator (DC brushed) is controlled by a separate micro-controller (LM629) that is connected to the main CPU via the digital I/O board. The micro-controllers use a bi-directional shared local I/O bus connecting them to the digital I/O board. Sensors are also connected to the analog and digital boards.

To optimize the performance of the system, we use the local trajectory generation of the micro-controllers. Coordinated motions are achieved by pre-computing trajectories for each joint, downloading them to the micro-controllers, and simultaneously starting the actuators. Whenever necessary, new set-points update the local trajectories while actuators are still moving. The status of the actuators is polled from the micro-controllers at a frequency of 30 Hz. This distributed architecture frees up the main processor for other tasks such as planning and sensor processing.

The on-board processor communicates with an external host via a wireless Ethernet at a maximum throughput of 1 MB/sec. However, we also use a wired 10Base2 line with a 10 MB/sec throughput for logging large sets of data.

C. An Object-Oriented Software Architecture

The criteria that guided the development of our software architecture is as follows. (i) We wanted to develop a robust and flexible hierarchical system that is easy to use, modify, and maintain. The hierarchical property provides various layers of abstraction with various levels of complexities at which we can work. (ii) We wanted to develop a reusable and extendible architecture to be able to easily migrate our coordination algorithms to different rover platforms. Building a modular system with reusable components reduces the amount of code that is written and speeds up the development process. (iii) We also wanted to develop a system that can separate the hardware dependencies to allow off-line testing and to simplify portability. Object-oriented design methodology provides the necessary tools to meet these criteria. The architecture described below is based on an earlier implementation developed in [6].

To achieve these goals, we developed a three-layered object-oriented system hierarchy in C++. At the lowest layer, we placed the system device drivers. The middle layer is the hardware abstraction layer which has a hierarchical structure and uses virtual mechanisms to talk to the hardware. Base classes represent the abstract and hardware independent functionality of a component. For example, a `Motor` class does not talk directly to a motor, but handles gear ratio conversion, actual motor position and velocity, discrepancy from the desired trajectory, etc. The actual implementation, which is hardware dependent, is carried out in a specialized class derived from the base class. To hide the hardware dependencies, parameter passing always uses base classes for types and not the derived classes. Off-the-shelf drivers, often written in C, are

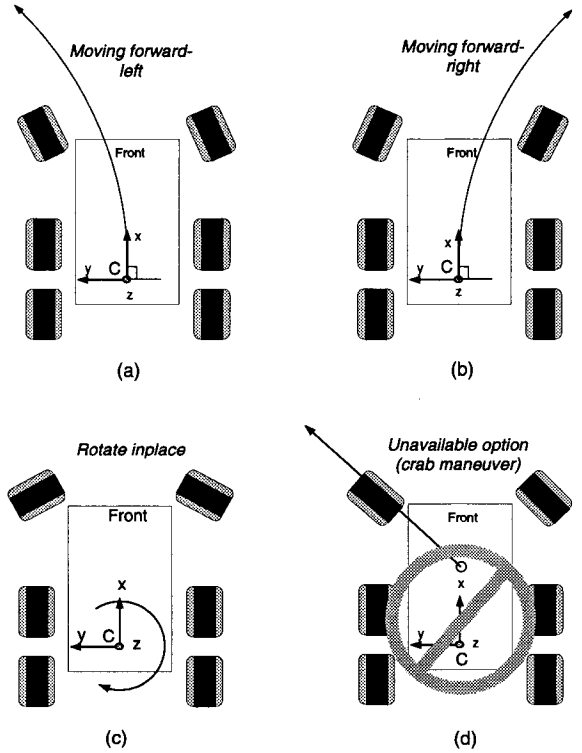

Fig. 2. The mobility of a six-wheel rover with two-wheel front steering (e.g. *Rocky 7*)

wrapped with C++ to implement these specialized classes which simplifies their interface and encapsulates their initializations. The third layer uses similar hierarchies to represent the various sub-systems such as `Arm`, `Mast`, `Vehicle`, etc. Higher level algorithms use classes from the middle and third layers to control the rover.

In addition to the classes that represent the system components, there are other hierarchies (some template-based) that handle data objects such as `Matrix`, `Vector`, `Image`, `Bits`, `Point`, etc, which are used in the system decomposition.

V. Control Strategy

Since we rely on frequent sensory feedback to achieve the rock acquisition and instrument placement operations, we do not attempt to develop accurate kinematic models of the vehicle and the arms. Rather, we rely on simplified approximate solutions that require minimal computation. Next, we present the models that were used in our approach.

A. Rover Mobility

Figure 2(a)-(c) shows the possible maneuvers that can be accomplished with a rocker-bogey mechanism having only two steerable wheels. Since the bogey wheels (back wheels) cannot change their orientation, all rover motions must occur along a circular path whose center lies along the axis of the non-steerable wheels (see Figure 3). Since there are two parallel axes for the non-steerable wheels, an equidistant axes is selected to minimizes the slippage and stress on the bogey wheels. Unless this type of vehicle moves in a straight line (infinite radius), slippage always occurs. Circular arcs of zero radius correspond to in-place rotations. This maneuver has the most slippage and results in large heading errors (unless heading sensors are used to compensate for these errors). This type of vehicle is unable to move laterally in a crab-like maneuver Figure 2(d) which would be desirable for our manipulation task. This restriction holds for all vehicles of this class independent of what algorithm is used to move the vehicle.

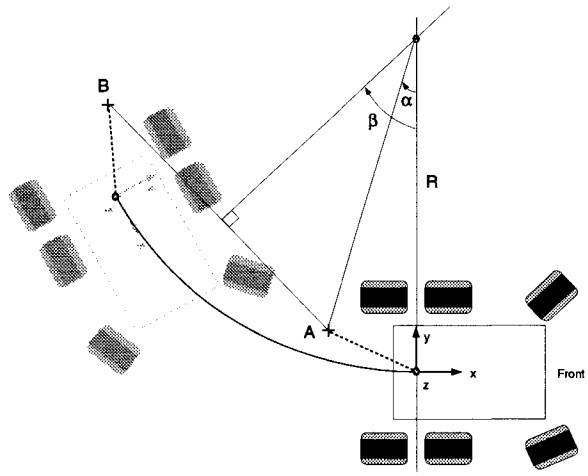

Fig. 3. Generating a single arc trajectory; point A is the optimal grasp location and point B is the goal

The above model is accurate when the rover is traversing level terrain. We define the vehicle's motion about point C which is the centroid of the bogey wheels. However, C is not a static point on the rover. It shifts as the rocker and bogey mechanism traverses rocky terrain. Nonetheless, it does not shift much since the rocker and bogey angles have limited ranges. Because we are only interested in the initial estimates of the wheels' motion parameters, we do not need to develop the complex three-dimensional kinematics for the mechanism. Rather, we use the flat terrain approximation to provide initial input to the wheel motors. We, then, constantly reevaluate these parameters based on visual feedback.

B. Driving the Vehicle to a Goal

Using this approximate geometrical approach, we would like to move the rover such that a specified point A on the rover reaches a goal point B specified in the two-dimensional terrain (see Figure 3). As long as the final rover orientation is unconstrained, a single arc motion of the rover is theoretically sufficient to drive point A to point B. This strategy forms the basis of the vehicle motions in our feedback system for the rock sample acquisition.

When using the manipulator arm which has two degrees of freedom, we have to rely on the vehicle's mobility mechanism to position the rover arm within 10 mm of the goal point. The workspace of the rover arm is merely the volume of a hollow hemisphere with a thick shell. The intersection of this workspace with the ground is an arc with a thickness of one inch. This is the reachable workspace of the

arm for rock acquisition. But because the end effector on that arm does not have a wrist roll, only a small area of about 40 mm in diameter (valid workspace region) can be used reliably to pick rocks. Consequently, the rover must drive toward the goal and precisely position the optimal arm location over the target rock. Once at the goal, the rover deploys its arm and picks the rock.

The rock sample acquisition does not require a specific rover orientation. However, orientation is constrained when placing an instrument onto a boulder. We discuss this in the next section.

C. Re-orienting the Vehicle while Driving

Another objective is to move a specified point A on the rover to a goal point B with a specified final orientation of the rover. For example, when placing an instrument on a boulder, the rover must approach a boulder in a given direction and be able to place the instrument along the surface normal of the area of interest. Once again we rely on the vehicle's mobility mechanism primarily for positioning and orienting the rover since the mast that carries the instruments has limited degrees of freedom.

The problem of moving and re-orienting the vehicle is over-constrained for a single arc trajectory. A minimum of two arcs is necessary to accomplish this task. There is an infinite number of arc pairs that can drive the rover to its destination with the proper final orientation (see Figure 4). When either arc length of the pair goes to zero, the rover does a rotate-in-place (Figure 2(c)) either at the beginning or at the end of the trajectory. However, we would like to minimize the rotate-in-place maneuvers since they have the most slippage and create the highest stresses on the bogey wheels. Besides, without a heading sensor, such as a sun sensor, a rotate-in-place yields the worst odometry causing us to lose track of the target. The criteria we use for path selection are: (i) minimize the sum of the arc lengths if the heading direction remains the same for the two arcs, or (ii) minimize the difference if the heading direction changes. This selects the shortest and smoothest path.

To validate the above strategies, we developed a Matlab simulation and visualization tool that displays possible path solutions as well as the selected path (Figure 4). The motion is animated as the rover traverses along the path toward its goal.

These strategies are used in the final approach to the goal since they assume that there are no obstacles between the rover and the target. In the next section, we will show how these motion strategies are updated by the visual tracker to achieve the overall task objectives.

D. Algorithm for Vision-Based Manipulation

Table I describes the algorithm that was used to acquire a small rock sample. Once the scientist selects the target rock, the selected point is transmitted back to the rover which uses stereo vision processing based on camera models to compute the three-dimensional location of the rock [14]. Using the x and y world coordinates of the rock, a single arc is computed as described above and the rover starts its traverse toward the target. The rover needs to be positioned such that the rock is inside the arm's valid workspace region. To compensate for the slippage in the unknown terrain, the approximate kinematics, and any other disturbances, we continuously acquire images along the path and re-evaluate the position of the rock. Although our control architecture allows us to perform this operation while the rover is in motion, we currently halt rover movement before taking new stereo images.

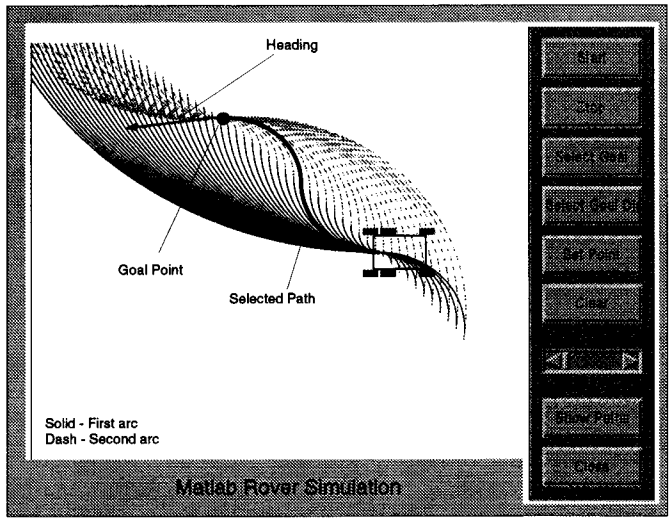

Fig. 4. Possible paths with optimal path selection (black line)

Thus, at 10 cm intervals, the rover stops to take new stereo images using the body navigation cameras. Using the vehicle's odometry, we compute a new estimated location of the target and a small window around that point is searched in an attempt to relocate the target. The search is done in the elevation map generated from the range image of the stereo pair. We search for the shape of the rock rather than its visual appearance. In particular, we assume that any target rock will be resting higher on the ground than its nearby surroundings, and lock in on the local elevation maximum as the new, refined 3D target point. Since we do not always have a dense elevation map, we linearly interpolate missing data from the range image before searching for the local maximum in the elevation map.

To compensate for the large errors in the odometry estimate, we also assume that our targets are visually distinct from the background sand, and use an intensity filter to focus attention in the elevation map. We have chosen this solution due to timeframe constraints. In fact, any pixel classification technique can be used instead of brightness, *e.g.*, [9]. If no range data is available, then no refinement is done, and the vehicle odometry is assumed to be correct. Further details on the visual tracking algorithm can be found in [4].

The rover tracks the target and continues to correct its motion trajectory until the arm's workspace region is centered over the target rock.